

```

        unsigned long EPCMgr;
        unsigned long ObjectClass;
        EPC[0] = header;                // header indicates 96-bit GTIN
        EPC[1] = objectType << 5 ;      // move objectType into high-order three bits
        EPC[1] |= partition << 2 ;      // move partition type into next three bits
        EPCMgr = (unsigned long)( UPC / 1000000 ); // extract company number / EPC manager
number
        // extract object class, discard check digit
        ObjectClass = (unsigned long)(UPC-(__int64)EPCMgr*1000000)/10;
        EPC[2] = (unsigned char)( EPCMgr / 16384 ); // shift and incorporate high bits
        EPC[3] = (unsigned char)( EPCMgr / 64 );    // shift and incorporate next bits
        EPC[4] = (unsigned char)( EPCMgr * 4 );     // shift and incorporate next bits
        ObjectClass += indicatorDigit*100000;
        EPC[5] = (unsigned char)( ObjectClass / 1024 ); // shift and incorporate high bits
        EPC[6] = (unsigned char)( ObjectClass / 4 );   // shift and incorporate next bits
        EPC[7] = (unsigned char)( ObjectClass * 64 ); // shift and incorporate last bits
        EPC[7] |= serialNumber[0]; // bits are already aligned
        EPC[8] = serialNumber[1]; // bits are already aligned
        EPC[9] = serialNumber[2]; // bits are already aligned
        EPC[10] = serialNumber[3]; // bits are already aligned
        EPC[11] = serialNumber[4]; // bits are already aligned
    }

```

APPENDIX C: Intel-Architecture Assembly Code

```

// First, sample code of calling procedure for function
lea     eax,[serialNumber]
push    eax
lea     eax,[EPC]
push    eax
push    1
push    3
push    5
push    10h
push    12h
push    556A9CD8h
call    UPC2EPC1
// Now actual function to convertUPC2EPC:
push    ebp
mov     ebp,esp
sub     esp,0D8h
push    ebx
push    esi
push    edi
lea     edi,[ebp-0D8h]
push    36h
pop     ecx
mov     eax,0CCCCCCCCh
rep
stos    dword ptr [edi]
8: unsigned long EPCMgr;
9: unsigned long ObjectClass;
// header indicates 96-bit GTIN
mov     eax,dword ptr [EPC]
mov     cl,byte ptr [header]
mov     byte ptr [eax],cl // move objectType into high-order three bits
movzx   eax,byte ptr [objectType]

```

```

shl     eax,5
mov     ecx,dword ptr [EPC]
mov     byte ptr [ecx+1],al      // move partition type into next three bits
movzx   eax,byte ptr [partition]
shl     eax,2
mov     ecx,dword ptr [EPC]
movzx   ecx,byte ptr [ecx+1]
or      ecx,eax
mov     eax,dword ptr [EPC]
mov     byte ptr [eax+1],cl      // extract company number / EPC manager number
push    0
push    0F4240h
push    dword ptr [ebp+0Ch]
push    dword ptr [UPC]
call    @ILT+340(__alldiv) (411159h)
mov     dword ptr [EPCMgr],eax // extract object class, discard check digit
mov     eax,dword ptr [EPCMgr]
mov     ecx,0F4240h
mul     eax,ecx
mov     ecx,dword ptr [UPC]
sub     ecx,eax
mov     eax,dword ptr [ebp+0Ch]
sbb     eax,edx
mov     eax,ecx
xor     edx,edx
push    0Ah
pop     ecx
div     eax,ecx
mov     dword ptr [ObjectClass],eax // shift and incorporate high bits
mov     eax,dword ptr [EPCMgr]
shr     eax,0Eh
mov     ecx,dword ptr [EPC]
mov     byte ptr [ecx+2],al      // shift and incorporate next bits
mov     eax,dword ptr [EPCMgr]
shr     eax,6
mov     ecx,dword ptr [EPC]
mov     byte ptr [ecx+3],al      // shift and incorporate next bits
mov     eax,dword ptr [EPCMgr]
shl     eax,2
mov     ecx,dword ptr [EPC]
mov     byte ptr [ecx+4],al
movzx   eax,byte ptr [indicatorDigit]
imul    eax,eax,186A0h
mov     ecx,dword ptr [ObjectClass]
add     ecx,eax
mov     dword ptr [ObjectClass],ecx // shift and incorporate high bits
mov     eax,dword ptr [ObjectClass]
shr     eax,0Ah
mov     ecx,dword ptr [EPC]
mov     byte ptr [ecx+5],al      // shift and incorporate next bits
mov     eax,dword ptr [ObjectClass]
shr     eax,2
mov     ecx,dword ptr [EPC]
mov     byte ptr [ecx+6],al      // shift and incorporate last bits
mov     eax,dword ptr [ObjectClass]
shl     eax,6

```